Lecture 10: October 31, 2019

Reading: §8 **Assignments:** Homework 3, due on November 8 at 11:59pm

1. Dictionary
   (a) Collection of key-value pairs
2. Creating dictionaries
   (a) Using `d = {}`
   (b) Using `d = dict()`
3. Methods for dictionaries
   (a) `k in D`: True if dictionary `D` has key `k`; else False
   (b) `D.keys()`: list of keys in `D`
   (c) `D.values()`: list of values in `D`
   (d) `D.items()`: list of tuples (key, value) in `D`
   (e) `D.get(k, d)`: if key `k` in `D`, return associated value; else return `d`
   (f) `del D[k]`: delete tuple with key `k` from `D`
   (g) `D.clear()`: delete all entries in `D`
4. Example: memos
   (a) Remember how slowly the recursive Fibonacci number program *rfib.py* ran? Here is a faster recursive version that uses memos [*rfibmemo.py*]
5. Sorting the dictionary
   (a) `sorted` sorts based on keys
6. Example: word frequency count
   (a) Unsorted [*wfc-1.py*]
   (b) Sorted alphabetically [*wfc-2.py*]
   (c) Sorted alphabetically, but dictionary order (note `key=str.lower()` in `sorted` [*wfc-2a.py*]
   (d) Sorted by frequency (treat `lambda x: x[1]` as an idiom to reference the *value* of the dictionary entry, not the *key*—to go from highest to lowest, replace `x[1]` with `-x[1]`) [*wfc-3.py*]
   (e) Sorted by frequency first, then alphabetically—note use of function `alphafreq(x)`; you can use any function here, and the parameter is the item [*wfc-4.py*]